User Manual for the Windows Matlab Version of BACC (Bayesian Analysis, Computation, and Communication)

William McCausland John J. Stevens

June 28, 2007

Important Information

About this Manual

This manual describes the software developed in connection with the project Bayesian Communication in the Social Sciences, Siddhartha Chib and John Geweke principle investigators. Acknowledgement in any resulting published work would be appreciated. This project was supported, in part, by Grants SBR-9600040 and SBR-9731037 from the National Science Foundation.

Help Keep This Software Free

The National Science Foundation supports this software and its continued development. It is important that we document the use of BACC. We respectfully request that all publications and working papers reporting the results of research using BACC software, include the following acknowledgement and reference:

> Computations reported in this paper were undertaken [in part] using the Bayesian Analysis, Computation and Communication software (`http://www.cirano.qc.ca/~bacc`) described in:
>
> > Geweke, J. (1999) "Using Simulation Methods for Bayesian Econometric Models: Inference, Development, and Communication" (with discussion and rejoinder), *Econometric Reviews* 18: 1-126.

BACC Software and Documentation

BACC software and documentation is available on the web at

```
http://www.cirano.qc.ca/~bacc
```

Please send any comments or questions to

```
bacc@econ.umn.edu
```

Contents

Important Information **3**

1 Getting Started with BACC **9**
1.1 Introduction . . . 9
1.2 Requirements . . . 9
1.3 Installation and Configuration . . . 9

2 Models **11**
2.1 Introduction . . . 11
2.1.1 Dimension parameters . . . 11
2.1.2 Unknown Quantities . . . 11
2.1.3 Known Quantities . . . 11
2.1.4 Data Generating Process . . . 12
2.1.5 Prior Distribution . . . 12
2.1.6 Creating a Model Instance . . . 12
2.1.7 Sampling Algorithms . . . 12
2.1.8 Marginal Likelihood . . . 12
2.2 The Normal Linear Model . . . 13
2.2.1 Dimension parameters . . . 13
2.2.2 Unknown Quantities . . . 13
2.2.3 Known Quantities . . . 13
2.2.4 Data Generating Process . . . 13
2.2.5 Priors . . . 13
2.2.6 Creating a Model Instance . . . 14
2.2.7 Sampling Algorithms . . . 14
2.3 The Seemingly Unrelated Regressions Model . . . 15
2.4 The I.I.D. Finite State Model . . . 16
2.4.1 Dimension parameters . . . 16
2.4.2 Unknown Quantities . . . 16
2.4.3 Known Quantities . . . 16
2.4.4 Data Generating Process . . . 16
2.4.5 Priors . . . 16
2.4.6 Creating a Model Instance . . . 16
2.4.7 Sampling Algorithms . . . 16

2.4.8 Marginal Likelihood . . . 17
2.5 The Non-Stationary First Order Markov Finite State Model . . . 18
2.5.1 Dimension parameters . . . 18
2.5.2 Unknown Quantities . . . 18
2.5.3 Known Quantities . . . 18
2.5.4 Data Generating Process . . . 18
2.5.5 Priors . . . 18
2.5.6 Creating a Model Instance . . . 19
2.5.7 Sampling Algorithms . . . 19
2.5.8 Marginal Likelihood . . . 19
2.6 The Stationary First Order Markov Finite State Model . . . 20
2.6.1 Dimension parameters . . . 20
2.6.2 Unknown Quantities . . . 20
2.6.3 Known Quantities . . . 20
2.6.4 Data Generating Process . . . 20
2.6.5 Priors . . . 20
2.6.6 Creating a Model Instance . . . 21
2.6.7 Sampling Algorithms . . . 21
2.7 A Univariate Linear Model with Normal Disturbances . . . 22
2.8 A Univariate Dichotomous Choice Model with Normal Disturbances . . . 24
2.9 A Univariate Censored Regression Model with Normal Disturbances . . . 25
2.10 A Univariate Observed Set Model with Normal Disturbances . . . 27
2.11 A Univariate Linear Model with Student t Disturbances . . . 29
2.12 A Univariate Dichotomous Choice Model with Student t Disturbances . . . 31
2.13 A Univariate Censored Regression Model with Student t Disturbances . . . 33
2.14 A Univariate Observed Set Model with Student t Disturbances . . . 35
2.15 A Univariate Linear Model with Finite Mixtures of Normals Disturbances . . . 37
2.16 An Autoregression Model . . . 39
2.17 An Autoregression Model with State Dependant Means . . . 41

3 BACC Commands **43**
3.1 Overview of BACC Commands . . . 43
3.2 Matlab Issues . . . 44
3.3 Detailed Description of Commands . . . 44
3.3.1 The `dirichletSim` Command . . . 45
3.3.2 The `expect1` Command . . . 46
3.3.3 The `expectN` Command . . . 49
3.3.4 The `extract` Command . . . 51
3.3.5 The `gammaSim` Command . . . 52
3.3.6 The `gaussianSim` Command . . . 53
3.3.7 The `listModelSpecs` Command . . . 54
3.3.8 The `listModels` Command . . . 55
3.3.9 The `miDelete` Command . . . 56
3.3.10 The `miLoad` Command . . . 57
3.3.11 The `miLoadAscii` Command . . . 58
3.3.12 The `miSave` Command . . . 59

3.3.13 The `miSaveAscii` Command . . . 60
3.3.14 The `minst` Command . . . 61
3.3.15 The `mlike` Command . . . 62
3.3.16 The `paretoSim` Command . . . 64
3.3.17 The `postfilter` Command . . . 65
3.3.18 The `postsim` Command . . . 66
3.3.19 The `postsimHM` Command . . . 67
3.3.20 The `priorRobust` Command . . . 68
3.3.21 The `priorfilter` Command . . . 70
3.3.22 The `priorsim` Command . . . 71
3.3.23 The `setseedconstant` Command . . . 72
3.3.24 The `setseedtime` Command . . . 73
3.3.25 The `weightedSmooth` Command . . . 74
3.3.26 The `wishartSim` Command . . . 76

4 A BACC Tutorial **77**
4.1 Loading Known Quantities . . . 78
4.2 Creating a Model Instance . . . 78
4.3 Simulating the Model . . . 78
4.4 Extracting and Deleting Model Instances . . . 79
4.5 Computing Marginal likelihoods . . . 80
4.6 Computing Moments of Functions of Interest . . . 80
4.7 Sensitivity to the Prior . . . 81
4.8 Kernel Smoothing of Simulations . . . 81
4.9 Plotting . . . 82
4.10 Saving and Loading Model Instances . . . 83
4.10.1 To/From Text Files . . . 83
4.10.2 To/From Binary Files . . . 83
4.11 Drawing from Various Distributions . . . 83
4.11.1 Dirichlet . . . 83
4.11.2 Gamma . . . 83
4.11.3 Gaussian (Multivariate Normal) . . . 84
4.11.4 Pareto . . . 84
4.11.5 Wishart (Multivariate Chi-squared) . . . 84

A Distributions **85**
A.1 The Dirichlet Distribution . . . 85
A.2 The Gamma Distribution . . . 85
A.3 The Normal Distribution . . . 86
A.4 The Pareto Distribution . . . 86
A.5 The Poisson Distribution . . . 86
A.6 The Wishart Distribution . . . 87

Bibliography **88**

Chapter 1

Getting Started with BACC

1.1 Introduction

The BACC software provides the user several commands for doing Bayesian analysis and communications. This document describes the function of these commands and their inputs and outputs. It also outlines some of the theory behind the commands, and provides references to the relevant literature.

The following versions of the BACC software and documentation are available:

- Windows Matlab
- Linux/Unix Matlab
- Windows R
- Linux/Unix R

This particular manual is for the Windows Matlab version of BACC.

1.2 Requirements

The Windows Matlab Version of the BACC software requires Matlab 5.0 or higher for Microsoft Windows 95 or higher.

1.3 Installation and Configuration

Follow these steps to install the Windows Matlab version of BACC.

1. Download the zip file `baccWinMatlab.zip` from the software page of the BACC website `http://www.cirano.qc.ca/~bacc`.

2. Unzip the zip file to the directory `C:\` using a standard unzipping utility such as PKZIP (available at `http://www.pkware.com`). Other freeware and shareware unzipping utilities are available on the web. This creates the directory `C:\BACC_Matlab` and fills it with all the necessary files. You may also install the software into an alternative directory, in which case the following steps need to be modified accordingly.

3. Start Matlab

4. Add the `C:\BACC_Matlab` directory to the Matlab path.

   Choose 'Set Path...' from the 'File' menu in Matlab, and add the directory to the path.

5. Change the Matlab working directory to `C:\BACC_Matlab`.

6. Install BACC using the command `installBACC`.

The Windows Matlab version of BACC is now installed. Follow these steps to run the sample program `testBACC.m`.

1. Start Matlab

2. In Matlab, change directory to the `Test` directory.

   ```
   >> cd C:\BACC_Matlab\Test
   ```

3. Run the test program.

   ```
   >> testBACC
   ```

Chapter 2

Models

2.1 Introduction

This document specifies the models currently supported by the BACC system. Each section following this one describes one of the supported models. Each model description is organized into subsections, following the pattern of this section. Appendix A gives the probability density and mass functions of the distributions used throughout the document.

2.1.1 Dimension parameters

All the quantities relevant to a model are treated as matrix valued. All matrix sizes are specified in terms of these dimension parameters. Examples of dimension parameters include the number of times a variable is observed, the number of individuals in a cross section, and the number of equations in a linear model. This subsection lists and describes the dimension parameters for a particular model.

2.1.2 Unknown Quantities

Unknown quantities are all the unobserved elements in a model. They include unknown parameters of the model, latent variables, and missing data. Separate sub-sub-sections discuss unknown quantities in each of these categories. Posterior simulation involves drawing these quantities from their posterior distribution; that is, their conditional distribution given known quantities.

2.1.3 Known Quantities

Known quantities are all the observed or user-specified values in a model. They include prior parameters, which index distributions within a family of prior distributions, and observed data. Separate sub-sub-sections discuss known quantities in both of these categories. The user of the BACC software must specify all the known quantities of a model in order to create an instance of the model.

2.1.4 Data Generating Process

This section specifies the conditional distribution of the endogenous observed data, given the unknown quantities and any observed data ancillary with respect to the unknown quantities.

2.1.5 Prior Distribution

This section specifies the marginal distribution of the unknown quantities, reflecting the user's prior beliefs about these quantities. These unknown quantities may or may not be independent. An example where they are not is a hierarchical prior, in which the prior density is expressed as the product of marginal densities of the "lowest level" unknowns and conditional densities of "higher level" unknowns given "lower level" unknowns.

2.1.6 Creating a Model Instance

This section gives all the model specific information a user requires to create a model instance. It specifies a short mnemonic label that identifies the model, the order in which the user gives the names to assign the unknown quantities, and the order in which to supply all the known quantities. To create a model instance, the user issues the `minst` command, with appropriate arguments (see section 3.3.14).

2.1.7 Sampling Algorithms

This section has brief descriptions of the algorithm used to generate samples of unknown quantities from their prior and posterior distributions. One subsection each concerns the prior distribution and the posterior distribution. For further details on the algorithms, the user should consult the internal (source code) documentation for the BACC system.

2.1.8 Marginal Likelihood

Where there is an analytical expression for the marginal likelihood in a model, this subsection provides that expression.

2.2 The Normal Linear Model

2.2.1 Dimension parameters

There are m equations, k covariate coefficients, and T observations of each variable.

2.2.2 Unknown Quantities

Unknown Parameters

There is a $k \times 1$ coefficient parameter β and a $m \times m$ precision parameter H.

2.2.3 Known Quantities

Prior Parameters

There is a $k \times 1$ coefficient mean vector $\underline{\beta}$, a $k \times k$ positive definite coefficient precision matrix $\underline{H}$, a precision degrees of freedom parameter $\underline{\nu} > \frac{m-1}{2}$, and a positive definite precision inverse scale parameter $\underline{S}$.

Data

There are m vectors of observations of dependent variables: $y_1, \ldots, y_m$. Each vector is $T \times 1$.

There are m matrices of observations of ancillary (with respect to unknown quantities) variables: $Z_1, \ldots, Z_m$. Each matrix is $T \times k$.

2.2.4 Data Generating Process

$$y \equiv \begin{bmatrix} y_1 \\ \vdots \\ y_m \end{bmatrix} = \begin{bmatrix} Z_1 \\ \vdots \\ Z_m \end{bmatrix} \beta + \begin{bmatrix} \epsilon_1 \\ \vdots \\ \epsilon_m \end{bmatrix} \equiv Z\beta + \epsilon$$

$$\begin{bmatrix} \epsilon_1 \\ \vdots \\ \epsilon_m \end{bmatrix} \mid \begin{bmatrix} Z_1 \\ \vdots \\ Z_m \end{bmatrix} \sim \mathrm{N}(0, H^{-1} \otimes I_T)$$

2.2.5 Priors

The unknown parameters β and H are a-priori independent, and have the following marginal distributions.

$$\beta \sim \mathrm{N}(\underline{\beta}, \underline{H}_{\beta}^{-1})$$

$$H \sim \mathrm{W}i(\underline{S}^{-1}, \underline{\nu})$$

When $m = 1$, the distribution of $\underline{S}H$ is chi-squared with $\underline{\nu}$ degrees of freedom.

2.2.6 Creating a Model Instance

The mnemonic label identifying the model is `nlm`.

Supply the names you wish to give the unknown quantities in the following order: first the name of β ("beta" for example) and then the name of H.

Supply the known quantities in the following order: $\underline{\beta}$, $\underline{H}_\beta$, $\underline{\nu}$, $\underline{S}$, Z, y.

2.2.7 Sampling Algorithms

Generating Prior Samples

Samples from the prior distribution of β and H are generated independently.

Generating Posterior Samples

The algorithm to generate samples from the posterior distribution $\beta, H|Z, y$ is a Gibbs sampling algorithm with two blocks, based on the following conditional posterior distributions.

$$\beta|H, y, Z \sim \mathrm{N}(\overline{\beta}, \overline{H}_\beta^{-1})$$

$$H|\beta, y, Z \sim \mathrm{W}(\overline{S}^{-1}, \overline{\nu})$$

where

$$\overline{H}_\beta = \underline{H}_\beta + Z'(H \otimes I_T)Z$$

$$\overline{\beta} = \overline{H}_\beta^{-1}[\underline{H}_\beta\underline{\beta} + Z'(H \otimes I_T)y]$$

$$\overline{S} = \underline{S} + [s_{ij}], s_{ij} = (y_i - Z_i\beta)'(y_i - Z_i\beta)$$

$$\overline{\nu} = \underline{\nu} + T$$

2.3 The Seemingly Unrelated Regressions Model

This is a special case of the Normal Linear Model with $m > 1$. Please see section 2.2.

2.4 The I.I.D. Finite State Model

2.4.1 Dimension parameters

There are m states, N individuals and T observation times.

2.4.2 Unknown Quantities

Unknown Parameters

There is a $1 \times m$ state probability vector π.

2.4.3 Known Quantities

Prior Parameters

There is a $1 \times m$ parameter $\underline{\alpha}$ indexing the prior distribution of π.

Data

There are state observations $s_{ti} \in \{1, \ldots, m\}$ for each individual i of N individuals and each observation period t of T periods.

$$S = \begin{bmatrix} s_{1,1} & \cdots & s_{N,1} \\ \vdots & \ddots & \vdots \\ s_{T,1} & \cdots & s_{N,T} \end{bmatrix}$$

2.4.4 Data Generating Process

Each observation s_{ti} is independently and identically distributed as follows.

$$\Pr(s_{ti} = s) = \pi_s \qquad s = 1, \ldots, m$$

2.4.5 Priors

$$\pi \sim \mathrm{Di}(\underline{\alpha})$$

2.4.6 Creating a Model Instance

The mnemonic label identifying the model is `iidfs`.

Supply the name you wish to give the unknown quantity π ("pi" for example).

Supply the known quantities in the following order: $\underline{\alpha}$, S.

2.4.7 Sampling Algorithms

Generating Prior Samples

Samples from the prior distribution of π are generated independently.

Generating Posterior Samples

In this model, the posterior distribution for π is the following familiar distribution.

$$\pi|S \sim \mathrm{Di}(\overline{\alpha})$$

where

$$\overline{\alpha} = \underline{\alpha} + n$$

$$n = [n_1 \cdots n_m]$$

and n_s is the number of observations for which $s_{ti} = s$. Posterior samples are drawn independently from this distribution.

2.4.8 Marginal Likelihood

The marginal likelihood is given by

$$p(S) = \frac{\Gamma(\sum_{i=1}^m \underline{\alpha}_i)}{\prod_{i=1}^m \Gamma(\underline{\alpha}_i)} \frac{\prod_{i=1}^m \Gamma(\overline{\alpha}_i)}{\Gamma(\sum_{i=1}^m \overline{\alpha}_i)}.$$

2.5 The Non-Stationary First Order Markov Finite State Model

2.5.1 Dimension parameters

There are m states, N individuals and T observation times.

2.5.2 Unknown Quantities

Unknown Parameters

There is a $1 \times m$ initial state probability vector π and an $m \times m$ Markov transition probability matrix P.

2.5.3 Known Quantities

Prior Parameters

The prior parameters are a $1 \times m$ vector $\underline{\alpha}_0$ indexing the prior distribution of π, and an $m \times m$ matrix $\underline{\alpha}$ indexing the prior distribution of P.

Data

There are state observations $s_{ti} \in \{1, \ldots, m\}$ for each individual i and each observation time t.

$$S = \begin{bmatrix} s_{11} & \cdots & s_{1N} \\ \vdots & \ddots & \vdots \\ s_{T1} & \cdots & s_{TN} \end{bmatrix}$$

2.5.4 Data Generating Process

The N observation sequences $\{s_{ti}\}_{t=1}^{T}$ are i.i.d., with each sequence being first order Markov. The initial distribution is π and the Markov transition matrix is P.

$$\Pr(s_{1i} = s) = \pi_s \qquad s = 1, \ldots, m$$

$$\Pr(s_{ti} = s' | s_{t-1,i} = s) = P_{ss'}$$

2.5.5 Priors

The m rows P_s of P and π are mutually independent, and have the following marginal distributions.

$$\pi \sim \mathrm{Di}(\underline{\alpha}_0)$$

$$P_s \equiv [P_{s1}, \ldots, P_{sm}] \sim \mathrm{D}i(\underline{\alpha}_s) \qquad s = 1, \ldots, m$$

$$\underline{\alpha}_0 \equiv \begin{bmatrix} \underline{\alpha}_{01} & \cdots & \underline{\alpha}_{0m} \end{bmatrix}$$

$$\underline{\alpha}_s \equiv \left[\begin{array}{ccc} \underline{\alpha}_{s1} & \cdots & \underline{\alpha}_{sm} \end{array} \right] \qquad s = 1, \ldots, m$$

2.5.6 Creating a Model Instance

The mnemonic label identifying the model is `nsfomfs`.

Supply the names you wish to give the unknown quantities in the following order: first the name of π ("pi" for example), and then the name of P.

Supply the known quantities in the following order: $\underline{\alpha}_0$, $\underline{\alpha}$, S.

2.5.7 Sampling Algorithms

Generating Prior Samples

Samples from the prior distributions of π and P are generated independently.

Generating Posterior Samples

In the posterior distribution $\pi, P|S$, the parameters π and P are conditionally independent, and their marginal posterior distributions are the following familiar distributions.

$$\pi|S \sim \mathrm{D}i(\overline{\alpha}_0)$$

$$P_s|S \sim \mathrm{D}i(\overline{\alpha}_s) \qquad s = 1, \ldots, m$$

where

$$\begin{aligned} \overline{\alpha}_0 &\equiv \underline{\alpha}_0 + n_0 \\ \overline{\alpha} &\equiv \underline{\alpha} + n \\ \overline{\alpha}_s &\equiv \left[\begin{array}{ccc} \overline{\alpha}_{s1} & \cdots & \overline{\alpha}_{sm} \end{array} \right] \\ n_0 &\equiv \left[\begin{array}{ccc} n_{01} & \cdots & n_{0m} \end{array} \right] \\ n &\equiv \left[\begin{array}{ccc} n_{11} & \cdots & n_{1m} \\ \vdots & \ddots & \vdots \\ n_{m1} & \cdots & n_{mm} \end{array} \right] \end{aligned}$$

where n_{0s} is the number of individuals starting in state s, and $n_{ss'}$ is the number of transitions from state s to state s' in the data.

Posterior samples are drawn independently from this distribution.

2.5.8 Marginal Likelihood

The marginal likelihood is available in closed form:

$$p(S) = \frac{\Gamma(\sum_{s=1}^m \underline{\alpha}_{0s})}{\prod_{s=1}^m \Gamma(\underline{\alpha}_{0s})} \frac{\prod_{s=1}^m \Gamma(\overline{\alpha}_{0s})}{\Gamma(\sum_{s=1}^m \overline{\alpha}_{0s})} \cdot \prod_{s=1}^m \left[\frac{\Gamma(\sum_{s'=1}^m \underline{\alpha}_{ss'})}{\prod_{s'=1}^m \Gamma(\underline{\alpha}_{ss'})} \frac{\prod_{s'=1}^m \Gamma(\overline{\alpha}_{ss'})}{\Gamma(\sum_{s'=1}^m \overline{\alpha}_{ss'})} \right]$$

2.6 The Stationary First Order Markov Finite State Model

2.6.1 Dimension parameters

There are m states, N individuals and T observation times.

2.6.2 Unknown Quantities

Unknown Parameters

There is an $m \times m$ Markov transition probability matrix P.

2.6.3 Known Quantities

Prior Parameters

The prior parameter is an $m \times m$ matrix $\underline{\alpha}$ indexing the prior distribution of P.

Data

There are state observations $s_{ti} \in \{1, \ldots, m\}$ for each individual i and each observation time t.

$$S = \begin{bmatrix} s_{11} & \cdots & s_{1N} \\ \vdots & \ddots & \vdots \\ s_{T1} & \cdots & s_{TN} \end{bmatrix}$$

2.6.4 Data Generating Process

The N observation sequences $\{s_{ti}\}_{t=1}^{T}$ are i.i.d., with each sequence being first order Markov with transition matrix P. The initial distribution vector is assumed to be the invariant distribution π for P.

$$\Pr(s_{1i} = s) = \pi_s \qquad s = 1, \ldots, m$$

$$\Pr(s_{ti} = s' | s_{t-1,i} = s) = P_{ss'}$$

where π is the left eigenvector of P corresponding to the eigenvalue $\lambda = 1$.

2.6.5 Priors

The m rows P_s of P are mutually independent, and have the following marginal distributions.

$$P_s \equiv [P_{s1}, \ldots, P_{sm}] \sim \mathrm{Di}(\underline{\alpha}_s) \qquad s = 1, \ldots, m$$

$$\underline{\alpha}_s \equiv \begin{bmatrix} \underline{\alpha}_{s1} & \cdots & \underline{\alpha}_{sm} \end{bmatrix} \qquad s = 1, \ldots, m$$

2.6.6 Creating a Model Instance

The mnemonic label identifying the model is `sfomfs`.

Supply the name you wish to give the unknown quantity P.

Supply the known quantities in the following order: $\underline{\alpha}$, S.

2.6.7 Sampling Algorithms

Generating Prior Draws

Samples from the prior distribution of P are generated independently.

Generating Posterior Draws

In this model, an independance Metropolis-Hastings chain is used to draw from the posterior distribution for P. The distribution $P^*|S$ of candidate draws is

$$P_s^*|S \sim \mathrm{D}i(\overline{\alpha}_s) \qquad s = 1, \ldots, m$$

where

$$\overline{\alpha} \equiv \underline{\alpha} + n$$

$$\overline{\alpha}_s \equiv \begin{bmatrix} \overline{\alpha}_{s1} & \cdots & \overline{\alpha}_{sm} \end{bmatrix}$$

$$n \equiv \begin{bmatrix} n_{11} & \cdots & n_{1m} \\ \vdots & \ddots & \vdots \\ n_{m1} & \cdots & n_{mm} \end{bmatrix}$$

where $n_{ss'}$ is the number of transitions from state s to state s' in the data.

The Hastings ratio for this block is given by

$$\prod_{i=1}^{N} \frac{\pi^*_{S_{1i}}}{\pi_{S_{1i}}}$$

2.7 A Univariate Linear Model with Normal Disturbances

The mnemonic label identifying the model is `n_ulm`.

Dimension Parameters

T	number of observations
K	number of covariates

Unknown Quantities

β	$(K \times 1)$	vector of covariate coefficients
h	(1×1)	precision of disturbance

Known Quantities

$\underline{\beta}$	$(K \times 1)$	prior mean of β
$\underline{H}_\beta$	$(K \times K)$	prior precision of β
$\underline{s}^2$	(1×1)	prior inverse scale of h
$\underline{\nu}$	(1×1)	prior degrees of freedom of h
X	$(T \times K)$	covariates
y	$(T \times 1)$	dependant variable

Data Generating Process

The observables y are given by

$$y = X\beta + u,$$

where u is a $T \times 1$ vector of i.i.d. normal disturbances, with $u_t|h \sim N(0, h^{-1})$:

$$p(u|X, \beta, h) = (2\pi)^{-T/2} h^{T/2} \exp(-hu'u/2).$$

Prior Distribution

The vectors β and h, together with X, are mutually independant. The covariate coefficient vector β has distribution $N(\underline{\beta}, \underline{H}_\beta)$:

$$p(\beta) = (2\pi)^{-K/2} |\underline{H}_\beta|^{1/2} \exp[-(\beta - \underline{\beta})' \underline{H}_\beta (\beta - \underline{\beta})/2].$$

The precision parameter h has a scaled chi-squared distribution, with $\underline{s}^2 h \sim \underline{\nu}$:

$$p(h) = 2^{-\underline{\nu}/2} \Gamma(\underline{\nu}/2)^{-1} (\underline{s}^2)^{\underline{\nu}/2} h^{(\underline{\nu}-2)/2} \exp(-\underline{s}^2 h/2).$$

Sampling Algorithm

See Example 3.4.1 in "Contemporary Bayesian Econometrics and Statistics," by John Geweke, at `http://www.cirano.qc.ca/~bacc/bacc2003/resources.html`

2.8 A Univariate Dichotomous Choice Model with Normal Disturbances

The mnemonic label identifying the model is `n_dichot`.

Dimension Parameters

T	number of observations
K	number of covariates

Unknown Quantities

β	$(K \times 1)$	vector of covariate coefficients
$\tilde{y}$	$(T \times 1)$	normal latent variables

Known Quantities

$\underline{\beta}$	$(K \times 1)$	prior mean of β
$\underline{H}_\beta$	$(K \times K)$	prior precision of β
X	$(T \times K)$	covariates
y	$(T \times 1)$	dependant variable

Data Generating Process

y is a $T \times 1$ vector with

$$y_t = \begin{cases} 0 & \tilde{y}_t \in (-\infty, 0) \\ 1 & \tilde{y}_t \in [0, \infty) \end{cases}$$

u is a $T \times 1$ vector of i.i.d. normal disturbances, with $u_t|h \sim N(0, h^{-1})$:

$$p(u|X, \beta, h) = (2\pi)^{-T/2} h^{T/2} \exp(-hu'u/2).$$

Prior Distribution

The covariate coefficient vector β has distribution $N(\underline{\beta}, \underline{H}_\beta)$:

$$p(\beta) = (2\pi)^{-K/2} |\underline{H}_\beta|^{1/2} \exp[-(\beta - \underline{\beta})' \underline{H}_\beta (\beta - \underline{\beta})/2].$$

Sampling Algorithm

See Example 3.4.1 in "Contemporary Bayesian Econometrics and Statistics," by John Geweke, at `http://www.cirano.qc.ca/~bacc/bacc2003/resources.html`

2.9 A Univariate Censored Regression Model with Normal Disturbances

The mnemonic label identifying the model is `n_censor`.

Dimension Parameters

T	number of observations
K	number of covariates

Unknown Quantities

β	$(K \times 1)$	vector of covariate coefficients
h	(1×1)	precision of student t distribution
$\tilde{y}$	$(T \times 1)$	normal latent variables

Known Quantities

$\underline{\beta}$	$(K \times 1)$	prior mean of β
$\underline{H}_\beta$	$(K \times K)$	prior precision of β
$\underline{s}^2$	(1×1)	prior inverse scale of h
$\underline{\nu}$	(1×1)	prior degrees of freedom of h
y_0	(1×1)	censoring threshold
X	$(T \times K)$	covariates
y	$(T \times 1)$	dependant variable

Data Generating Process

y is a $T \times 1$ vector with

$$y_t = \begin{cases} 0 & \tilde{y}_t \in (-\infty, 0) \\ 1 & \tilde{y}_t \in [0, \infty) \end{cases}$$

u is a $T \times 1$ vector of i.i.d. normal disturbances, with $u_t|h \sim N(0, h^{-1})$:

$$p(u|X, \beta, h) = (2\pi)^{-T/2} h^{T/2} \exp(-hu'u/2).$$

Prior Distribution

The vectors β, h are mutually independant. The covariate coefficient vector β has distribution $N(\underline{\beta}, \underline{H}_\beta)$:

$$p(\beta) = (2\pi)^{-K/2} |\underline{H}_\beta|^{1/2} \exp[-(\beta - \underline{\beta})' \underline{H}_\beta (\beta - \underline{\beta})/2].$$

The precision parameter h has a scaled chi-squared distribution, with $\underline{s}^2 h \sim \underline{\nu}$:

$$p(h) = 2^{-\underline{\nu}/2}\Gamma(\underline{\nu}/2)^{-1}(\underline{s}^2)^{\underline{\nu}/2}h^{(\underline{\nu}-2)/2}\exp(-\underline{s}^2 h/2).$$

Sampling Algorithm

See Example 3.4.1 in "Contemporary Bayesian Econometrics and Statistics," by John Geweke, at `http://www.cirano.qc.ca/~bacc/bacc2003/resources.html`

2.10 A Univariate Observed Set Model with Normal Disturbances

The mnemonic label identifying the model is `n_set`.

Dimension Parameters

T	number of observations
K	number of covariates

Unknown Quantities

β	$(K \times 1)$	vector of covariate coefficients
h	(1×1)	precision of student t distribution
$\tilde{y}$	$(T \times 1)$	normal latent variables

Known Quantities

$\underline{\beta}$	$(K \times 1)$	prior mean of β
$\underline{H}_\beta$	$(K \times K)$	prior precision of β
$\underline{s}^2$	(1×1)	prior inverse scale of h
$\underline{\nu}$	(1×1)	prior degrees of freedom of h
X	$(T \times K)$	covariates
c	$(T \times 1)$	lower bound of observed interval
d	$(T \times 1)$	upper bound of observed interval

Data Generating Process

y is a $T \times 1$ vector with

$$y_t = \begin{cases} 0 & \tilde{y}_t \in (-\infty, 0) \\ 1 & \tilde{y}_t \in [0, \infty) \end{cases}$$

u is a $T \times 1$ vector of i.i.d. normal disturbances, with $u_t|h \sim N(0, h^{-1})$:

$$p(u|X, \beta, h) = (2\pi)^{-T/2} h^{T/2} \exp(-hu'u/2).$$

Prior Distribution

The vectors β, h are mutually independant. The covariate coefficient vector β has distribution $N(\underline{\beta}, \underline{H}_\beta)$:

$$p(\beta) = (2\pi)^{-K/2} |\underline{H}_\beta|^{1/2} \exp[-(\beta - \underline{\beta})' \underline{H}_\beta (\beta - \underline{\beta})/2].$$

The precision parameter h has a scaled chi-squared distribution, with $\underline{s}^2 h \sim \underline{\nu}$:

$$p(h) = 2^{-\underline{\nu}/2}\Gamma(\underline{\nu}/2)^{-1}(\underline{s}^2)^{\underline{\nu}/2}h^{(\underline{\nu}-2)/2}\exp(-\underline{s}^2 h/2).$$

Sampling Algorithm

See Example 3.4.1 in "Contemporary Bayesian Econometrics and Statistics," by John Geweke, at `http://www.cirano.qc.ca/~bacc/bacc2003/resources.html`

2.11 A Univariate Linear Model with Student t Disturbances

The mnemonic label identifying the model is `t_ulm`.

Dimension Parameters

T	number of observations
K	number of covariates

Unknown Quantities

β	$(K \times 1)$	vector of covariate coefficients
h	(1×1)	precision of Student t distribution
$\tilde{h}$	$(T \times 1)$	time varying latent precision variable
λ	(1×1)	degrees of freedom of Student t distribution

Known Quantities

$\underline{\beta}$	$(K \times 1)$	prior mean of β
$\underline{H}_\beta$	$(K \times K)$	prior precision of β
$\underline{s}^2$	(1×1)	prior inverse scale of h
$\underline{\nu}$	(1×1)	prior degrees of freedom of h
$\underline{\lambda}$	(1×1)	prior mean of λ
X	$(T \times K)$	covariates
y	$(T \times 1)$	dependant variable

Data Generating Process

The observables y are given by

$$y = X\beta + u,$$

where u is a $T \times 1$ vector of independant Student t disturbances, with $u_t|h, X \sim t(0, h^{-1}, \lambda)$. Conditioning on the latent $\tilde{h}$ gives $u_t|h, \tilde{h} \sim N(0, (h\tilde{h}_t)^{-1})$:

$$p(u|X, \beta, \tilde{h}, h) = (2\pi)^{-T/2} h^{T/2} \prod_{t=1}^{T} \tilde{h}_t^{1/2} \exp(-h\tilde{h}_t u_t^2/2).$$

See Section 4.8 in "Contemporary Bayesian Econometrics and Statistics," by John Geweke, at `http://www.cirano.qc.ca/~bacc/bacc2003/resources.html` for details.

Prior Distribution

The vectors β, h, and $(\tilde{h}, \lambda)$, together with X, are mutually independant. The covariate coefficient vector β has distribution $N(\underline{\beta}, \underline{H}_\beta)$:

$$p(\beta) = (2\pi)^{-K/2} |\underline{H}_\beta|^{1/2} \exp[-(\beta - \underline{\beta})' \underline{H}_\beta (\beta - \underline{\beta})/2].$$

The precision parameter h has a scaled chi-squared distribution, with $\underline{s}^2 h \sim \underline{\nu}$:

$$p(h) = 2^{-\underline{\nu}/2} \Gamma(\underline{\nu}/2)^{-1} (\underline{s}^2)^{\underline{\nu}/2} h^{(\underline{\nu}-2)/2} \exp(-\underline{s}^2 h/2).$$

The time varying latent precision parameters $\tilde{h}$ are i.i.d. scaled chi-squared variates, with $\lambda \tilde{h}_t \sim \chi^2(\lambda)$:

$$p(\tilde{h}|\lambda) = [2^{\lambda/2} \Gamma(\lambda/2)]^{-T} \lambda^{T\lambda/2} \prod_{t=1}^{T} \tilde{h}_t^{(\lambda-2)/2} \exp(-\lambda \tilde{h}/2)$$

The degrees of freedom parameter λ is distributed $\exp(\underline{\lambda})$:

$$p(\lambda) = \underline{\lambda}^{-1} \exp(-\lambda/\underline{\lambda}).$$

Sampling Algorithm

See Section 4.8 in "Contemporary Bayesian Econometrics and Statistics," by John Geweke, at `http://www.cirano.qc.ca/~bacc/bacc2003/resources.html`

2.12 A Univariate Dichotomous Choice Model with Student t Disturbances

The mnemonic label identifying the model is `t_dichot`.

Dimension Parameters

T	number of observations
K	number of covariates

Unknown Quantities

β	$(K \times 1)$	vector of covariate coefficients
$\tilde{h}$	$(T \times 1)$	time varying latent precision variable
λ	(1×1)	degrees of freedom of Student t distribution
$\tilde{y}$	$(T \times 1)$	normal latent variables

Known Quantities

$\underline{\beta}$	$(K \times 1)$	prior mean of β
$\underline{H}_\beta$	$(K \times K)$	prior precision of β
$\underline{\lambda}$	(1×1)	prior mean of λ
X	$(T \times K)$	covariates
y	$(T \times 1)$	dependant variable

Data Generating Process

y is a $T \times 1$ vector with

$$y_t = \begin{cases} 0 & \tilde{y}_t \in (-\infty, 0) \\ 1 & \tilde{y}_t \in [0, \infty) \end{cases}$$

u is a $T \times 1$ vector of i.i.d. normal disturbances, with $u_t|h \sim N(0, h^{-1})$:

$$p(u|X, \beta, h) = (2\pi)^{-T/2} h^{T/2} \exp(-hu'u/2).$$

Prior Distribution

The covariate coefficient vector β has distribution $N(\underline{\beta}, \underline{H}_\beta)$:

$$p(\beta) = (2\pi)^{-K/2} |\underline{H}_\beta|^{1/2} \exp[-(\beta - \underline{\beta})' \underline{H}_\beta (\beta - \underline{\beta})/2].$$

Sampling Algorithm

See Example 3.4.1 in "Contemporary Bayesian Econometrics and Statistics," by John Geweke, at `http://www.cirano.qc.ca/~bacc/bacc2003/resources.html`

2.13 A Univariate Censored Regression Model with Student t Disturbances

The mnemonic label identifying the model is `t_censor`.

Dimension Parameters

T	number of observations
K	number of covariates

Unknown Quantities

β	$(K \times 1)$	vector of covariate coefficients
h	(1×1)	precision of student t distribution
$\tilde{h}$	$(T \times 1)$	time varying latent precision variable
λ	(1×1)	degrees of freedom of Student t distribution
$\tilde{y}$	$(T \times 1)$	normal latent variables

Known Quantities

$\underline{\beta}$	$(K \times 1)$	prior mean of β
$\underline{H}_\beta$	$(K \times K)$	prior precision of β
$\underline{s}^2$	(1×1)	prior inverse scale of h
$\underline{\nu}$	(1×1)	prior degrees of freedom of h
$\underline{\lambda}$	(1×1)	prior mean of λ
y_0	(1×1)	censoring threshold
X	$(T \times K)$	covariates
y	$(T \times 1)$	dependant variable

Data Generating Process

y is a $T \times 1$ vector with

$$y_t = \begin{cases} 0 & \tilde{y}_t \in (-\infty, 0) \\ 1 & \tilde{y}_t \in [0, \infty) \end{cases}$$

u is a $T \times 1$ vector of i.i.d. normal disturbances, with $u_t|h \sim N(0, h^{-1})$:

$$p(u|X, \beta, h) = (2\pi)^{-T/2} h^{T/2} \exp(-hu'u/2).$$

Prior Distribution

The vectors β, h are mutually independant. The covariate coefficient vector β has distribution $N(\underline{\beta}, \underline{H}_\beta)$:

$$p(\beta) = (2\pi)^{-K/2} |\underline{H}_\beta|^{1/2} \exp[-(\beta - \underline{\beta})' \underline{H}_\beta (\beta - \underline{\beta})/2].$$

The precision parameter h has a scaled chi-squared distribution, with $\underline{s}^2 h \sim \underline{\nu}$:

$$p(h) = 2^{-\underline{\nu}/2} \Gamma(\underline{\nu}/2)^{-1} (\underline{s}^2)^{\underline{\nu}/2} h^{(\underline{\nu}-2)/2} \exp(-\underline{s}^2 h/2).$$

Sampling Algorithm

See Example 3.4.1 in "Contemporary Bayesian Econometrics and Statistics," by John Geweke, at `http://www.cirano.qc.ca/~bacc/bacc2003/resources.html`

2.14 A Univariate Observed Set Model with Student t Disturbances

The mnemonic label identifying the model is `t_set`.

Dimension Parameters

T	number of observations
K	number of covariates

Unknown Quantities

β	$(K \times 1)$	vector of covariate coefficients
h	(1×1)	precision of Student t distribution
$\tilde{h}$	$(T \times 1)$	time varying latent precision variable
λ	(1×1)	degrees of freedom of Student t distribution
$\tilde{y}$	$(T \times 1)$	normal latent variables

Known Quantities

$\underline{\beta}$	$(K \times 1)$	prior mean of β
$\underline{H}_\beta$	$(K \times K)$	prior precision of β
$\underline{s}^2$	(1×1)	prior inverse scale of h
$\underline{\nu}$	(1×1)	prior degrees of freedom of h
$\underline{\lambda}$	(1×1)	prior mean of λ
X	$(T \times K)$	covariates
c	$(T \times 1)$	lower bound of observed interval
d	$(T \times 1)$	upper bound of observed interval

Data Generating Process

y is a $T \times 1$ vector with

$$y_t = \begin{cases} 0 & \tilde{y}_t \in (-\infty, 0) \\ 1 & \tilde{y}_t \in [0, \infty) \end{cases}$$

u is a $T \times 1$ vector of i.i.d. normal disturbances, with $u_t|h \sim N(0, h^{-1})$:

$$p(u|X, \beta, h) = (2\pi)^{-T/2} h^{T/2} \exp(-hu'u/2).$$

Prior Distribution

The vectors β, h are mutually independant. The covariate coefficient vector β has distribution $N(\underline{\beta}, \underline{H}_\beta)$:

$$p(\beta) = (2\pi)^{-K/2} |\underline{H}_\beta|^{1/2} \exp[-(\beta - \underline{\beta})' \underline{H}_\beta (\beta - \underline{\beta})/2].$$

The precision parameter h has a scaled chi-squared distribution, with $\underline{s}^2 h \sim \underline{\nu}$:

$$p(h) = 2^{-\underline{\nu}/2} \Gamma(\underline{\nu}/2)^{-1} (\underline{s}^2)^{\underline{\nu}/2} h^{(\underline{\nu}-2)/2} \exp(-\underline{s}^2 h/2).$$

Sampling Algorithm

See Example 3.4.1 in "Contemporary Bayesian Econometrics and Statistics," by John Geweke, at `http://www.cirano.qc.ca/~bacc/bacc2003/resources.html`

2.15 A Univariate Linear Model with Finite Mixtures of Normals Disturbances

The mnemonic label identifying the model is `fmn_ulm`.

Dimension Parameters

T	number of observations
K	number of covariates
m	number of mixture components (or states)

Unknown Quantities

γ	$(m + K \times 1)$	vector of state means and covariate coefficients
h	(1×1)	constant multiplicative precision component
$\mathbf{h}$	$(m \times 1)$	state dependant multiplicative precision component
$\tilde{s}$	$(T \times 1)$	time varying latent discrete state
π	$(1 \times m)$	state probabilities

Known Quantities

$\underline{h}_\alpha$	(1×1)	prior precision parameter for state means
$\underline{\beta}$	$(K \times 1)$	prior mean of covariate coefficients
$\underline{H}_\beta$	$(K \times K)$	prior precision of covariate coefficients
$\underline{s}^2$	(1×1)	prior inverse scale of h
$\underline{\nu}$	(1×1)	prior degrees of freedom of h
m	(1×1)	number of states
$\underline{\nu}_*$	(1×1)	degrees of freedom parameter for state precisions
r	(1×1)	Dirichlet parameter for state probabilities
X	$(T \times K)$	covariates
y	$(T \times 1)$	dependant variable

Data Generating Process

The observables y are given by

$$y = X\beta + u,$$

where u is a $T \times 1$ vector of independant discrete normal mixture disturbances, with $u_t|h, \pi, \alpha, \mathbf{h}, X$ given by:

$$p(u_t|h, \pi, \alpha, \mathbf{h}, X) = (2\pi)^{-1/2} h^{1/2} \sum_{j=1}^{m} \pi_j h_j^{1/2} \exp[-h \cdot h_j (u_t - \alpha_j)^2/2]$$

Conditioning on the latent states gives

$$p(u_t|h, \alpha, \mathbf{h}, X) = (2\pi)^{-1/2} h^{1/2} h_{\tilde{s}_t}^{1/2} \exp[-h \cdot h_{\tilde{s}_t} (u_t - \alpha_{\tilde{s}_t})^2/2].$$

See Section 4.8 in "Contemporary Bayesian Econometrics and Statistics," by John Geweke, at `http://www.cirano.qc.ca/~bacc/bacc2003/resources.html` for details.

Prior Distribution

The vectors (γ, h), $\mathbf{h}$, and $(\tilde{s}, \pi)$, together with X, are mutually independant. The γ parameter vertically stacks the parameters α and β, where α is the $m \times 1$ vector of state dependant means and β is the $K \times 1$ vector of covariate coefficients. They are independent, with $\alpha|h \sim N(0, (\underline{h}_\alpha h)^{-1})$ and $\beta \sim N(\underline{\beta}, \underline{H}_\beta)$:

$$\begin{aligned} p(\gamma|h) = p(\alpha|h) \cdot p(\beta) &= (2\pi)^{-m/2} (\underline{h}_\alpha h)^{m/2} \exp(-\underline{h}_\alpha h \alpha' \alpha/2) \\ &\cdot \quad (2\pi)^{-K/2} |\underline{H}_\beta|^{1/2} \exp[-(\beta - \underline{\beta})' \underline{H}_\beta (\beta - \underline{\beta})/2]. \end{aligned}$$

The precision parameter h has a scaled chi-squared distribution, with $\underline{s}^2 h \sim \underline{\nu}$:

$$p(h) = 2^{-\underline{\nu}/2} \Gamma(\underline{\nu}/2)^{-1} (\underline{s}^2)^{\underline{\nu}/2} h^{(\underline{\nu}-2)/2} \exp(-\underline{s}^2 h/2).$$

The state dependant precisions h_j are i.i.d., with $\underline{\nu}_. h_j \sim \chi^2(\underline{\nu}_.)$:

$$p(\mathbf{h}) = 2^{m\underline{\nu}_.} \Gamma(\underline{\nu}_./2) (\underline{\nu}_.^2)^{m\underline{\nu}_./2} \prod_{j=1}^{m} h_j^{(-\underline{\nu}_.-2)/2} \exp(-\underline{\nu}_. h_j/2).$$

The latent states are i.i.d., with the probability $\Pr[s_t = j]$ given by π_j, for $j = 1, \ldots, m$:

$$p(\tilde{s}|\pi) = \prod_{t=1}^{T} \pi_{\tilde{s}_t}$$

The vector π of probabilities is distributed Dirichlet$(r, \ldots, r)$:

$$p(\pi) = \Gamma(mr) \Gamma(r)^{-m} \prod_{j=1}^{m} \pi_j^{r-1}.$$

Sampling Algorithm

See section 4.8 in "Contemporary Bayesian Econometrics and Statistics," by John Geweke, at `http://www.cirano.qc.ca/~bacc/bacc2003/resources.html`

2.16 An Autoregression Model

The mnemonic label identifying the model is `AR`.

Dimension Parameters

T	number of observations
K	number of covariates
p	autoregressive order

Unknown Quantities

β	$(K \times 1)$	covariate coefficient vector
h	(1×1)	residual precision
ϕ	$(p \times 1)$	vector of autoregression coefficients

Known Quantities

$\bar{\beta}$	$(K \times 1)$	prior mean of β
$\bar{H}_\beta$	$(K \times K)$	prior precision of β
$\bar{\nu}$	(1×1)	prior degrees of freedom of h
$\bar{s}^2$	(1×1)	prior inverse scale of h
$\bar{\phi}$	$(p \times 1)$	prior mean of ϕ before truncation
$\bar{H}_\phi$	$(p \times p)$	prior precision of ϕ before truncation
X	$(T \times K)$	covariates
y	$(T \times 1)$	dependant variable

Data Generating Process

The data generating process is given by

$$y_t = \beta' x_t + \epsilon_t$$

where x_t is the t'th row of X, as a column vector,

$$\epsilon_t = \sum_{i=1}^{p} \phi_i \epsilon_{t-i} + u_t,$$

and

$$u_t \sim \text{i.i.d. } \mathrm{N}(0, h^{-1})$$

Prior Distribution

The unknowns are a-priori independent and have the following distributions.

$$\beta \sim \mathrm{N}(\bar{\beta}, \bar{H}_{\beta}^{-1})$$

$$\bar{s}^2 h \sim \chi^2(\bar{\nu})$$

The prior for ϕ is obtained by truncating the following density to the region for which y is stationary.

$$\phi \sim \mathrm{N}(\bar{\phi}, \bar{H}_{\phi}^{-1})$$

Sampling Algorithm

The sampling algorithm for prior simulation features three blocks, each making independent draws from the prior distribution of one of the unknown quantities. The sampling algorithm for posterior simulation features three blocks, each making draws from the conditional posterior distribution of one of the unknown quantities.

2.17 An Autoregression Model with State Dependant Means

The mnemonic label identifying the model is `Hamilton`.

Dimension Parameters

T	number of observations
K	number of covariates
p	autoregressive order
m	number of states

Unknown Quantities

γ	$((m+K)\times 1)$	vertical stack of alpha and beta
h	(1×1)	residual precision
ϕ	$(p\times 1)$	vector of autoregression coefficients
P	$(m\times m)$	state transition probability matrix
s	$(T\times 1)$	latent states
f	$(T\times m)$	filter probabilities

Known Quantities

$\bar{\gamma}$	$((m+K)\times 1)$	prior mean of γ
$\bar{H}_\gamma$	$((m+K)\times(m+K))$	prior precision of γ
$\bar{\nu}$	(1×1)	prior degrees of freedom of h
$\bar{s}^2$	(1×1)	prior inverse scale of h
$\bar{\phi}$	$(p\times 1)$	prior mean of ϕ before truncation
$\bar{H}_\phi$	$(p\times p)$	prior precision of ϕ before truncation
$\bar{A}$	$(m\times m)$	parameters of prior for P
X	$(T\times K)$	covariates
y	$(T\times 1)$	dependant variable

Data Generating Process

The data generating process is given by

$$y_t = \alpha_{s_t} + \beta' x_t + \epsilon_t$$

where x_t is the t'th row of X, as a column vector, and α $(m \times 1)$ and β $(K \times 1)$ are obtained by partitioning γ,

$$\epsilon_t = \sum_{i=1}^{p} \phi_i \epsilon_{t-i} + u_t,$$

and

$$u_t \sim \text{i.}i.d.\ N(0, h^{-1})$$

Prior Distribution

The unknowns are a-priori independent and have the following distributions.

$$\gamma \sim \mathrm{N}(\bar{\gamma}, \bar{H_\gamma}^{-1})$$

$$\bar{s}^2 h \sim \chi^2(\bar{\nu})$$

The prior for ϕ is obtained by truncating the following density to the region for which y is stationary.

$$\phi \sim \mathrm{N}(\bar{\phi}, \bar{H}_\phi^{-1})$$

$$(P_{i1}, \ldots, P_{im}) \sim \text{i.i.d. Di}(\bar{A}_{i1}, \ldots, \bar{A}_{im})$$

$$\Pr[s_t = j | s_{t-1} = i] = P_{ij}$$

The unknown quantity f gives, for each observation time t, the state probabilities at t given previous states, previous values of the observed variables, and the other unknown quantities. It is not a primitive unknown quantity, and it is included to give the user access to filtered probabilities.

Sampling Algorithm

The sampling algorithm for prior simulation features five blocks. Four blocks make independent draws from the prior distributions of γ, h, ϕ and P. The fifth makes draws from the distribution $s|P$. The sampling algorithm for posterior simulation features five blocks, each making draws from the conditional posterior distribution of one of the unknown quantities.

Chapter 3

BACC Commands

3.1 Overview of BACC Commands

The following is a list of BACC commands with brief descriptions.

`dirichletSim`	Generates a sample from a multiple Dirichlet distribution.
`expect1`	Calculates, for a weighted random sample, the sample mean and standard deviation, estimates of the numerical standard error for the mean, and estimates of the relative numerical efficiency.
`expectN`	Calculates combined sample means, with numerical standard errors, for a set of different weighted random samples, and tests for the equality of their individual population means.
`extract`	Returns simulation matrices for a model instance.
`gammaSim`	Generates a sample from a gamma distribution.
`gaussianSim`	Generates a sample from a Gaussian distribution.
`listModelSpecs`	Lists all available model specifications (e.g. nlm, poisson).
`listModels`	Lists all open model instances.
`miDelete`	Closes without saving a (or all) model instances.
`miLoad`	Loads a model instance stored in a binary file.
`miLoadAscii`	Loads a model instance stored in a text file.
`miSave`	Saves a model instance in a binary file.
`miSaveAscii`	Saves a model instance in a text file.
`minst`	Creates an instance of a particular model specification.

`mlike`	Computes various estimates of the marginal likelihood for a model instance, with numerical standard errors.
`paretoSim`	Generates a sample from a Pareto distribution.
`postfilter`	Filters out previously generated draws from the posterior simulation matrix of a given model instance.
`postsim`	Generates or appends to the posterior simulation matrix of a given model instance.
`postsimHM`	Generates or appends to the posterior HM simulation matrix of a given model instance.
`priorRobust`	Calculates upper and lower bounds on the mean of a posterior function of interest, as the prior distribution is varied from its original specification.
`priorfilter`	Filters out previously generated draws from the prior simulation matrix of a given model instance.
`priorsim`	Generates or appends to the prior simulation matrix of a given model instance.
`setseedconstant`	Sets the seeds of the random number generators to a constant value.
`setseedtime`	Sets the seeds of the random number generators to the number of seconds since the beginning of 1970.
`weightedSmooth`	Estimates a univariate density function for a weighted random sample, using a kernel smoothing algorithm adapted to weighted samples.
`wishartSim`	Generates a sample from a Wishart distribution.

3.2 Matlab Issues

Help is available within Matlab for BACC commands. Type `help` *commandName* at the Matlab prompt, or `help BACC` for a list of BACC commands.

3.3 Detailed Description of Commands

Each BACC command is described in detail in one of the following sections.

3.3.1 The `dirichletSim` Command

Description

Generates a sample from a multiple Dirichlet distribution.

Usage

```
sample = dirichletSim(A, n);
```

Inputs

`A`	m by K matrix: Dirichlet parameters
`n`	Integer: number of draws to generate

Outputs

`sample`	n by nK matrix: sample generated from multiple Dirichlet distribution

See Also

`paretoSim`, `gaussianSim`, `gammaSim`, `wishartSim`.

Example

```
A = [1.0,2.0,3.0;4.0,5.0,6.0];

sample = dirichletSim(A, 1000);
```

Details

The sample consists of n draws. Each of the n draws of the sample is an m by K matrix with independent rows. Each row has a Dirichlet distribution with parameters given by the corresponding row of A.

The result is given as a n by mK matrix, and each column gives a draw in column major order. See Appendix A for the parameterization of the Dirichlet distribution.

3.3.2 The `expect1` Command

Description

Calculates, for a weighted random sample, the sample mean and standard deviation, estimates of the numerical standard error for the mean, and estimates of the relative numerical efficiency.

Usage

```
[mean, std, nse, rne] = expect1(logWeight, sample);
[mean, std, nse, rne] = expect1(logWeight, sample, taper);
```

Inputs

`logWeight`	Vector of length M: log sample weights
`sample`	Vector of length M: sample of scalar draws
`taper`	Vector of length K: taper half-widths (optional)

Outputs

`mean`	Real scalar: weighted sample mean
`std`	Real scalar: weighted sample standard deviation
`nse`	Vector of length $K+1$: estimated numerical standard errors
`rne`	Vector of length $K+1$: estimated relative numerical efficiency

See Also

`expectN`, `priorRobust`.

Example

```
% Use default taper values
[mean, std, nse, rne] = expect1(lw, z);
% Use alternate taper values
taper = [4.0,8.0];
[mean, std, nse, rne] = expectN(lw, z, taper);
```

Details

Let $z = (z_1, \ldots, z_M)$ be the sample and $(\log w_1, \ldots, \log w_M)$ be the vector of log weights. Let $\lambda = (\lambda_1, \ldots, \lambda_K)$ be the vector of half-widths. The sample is broken into T groups of size $J \equiv M \operatorname{div} T$ and the last $M \bmod T$ elements are ignored. Thus $M_{use} \equiv JT$ elements are used.

The sample mean and standard deviation are calculated as follows:

$$\bar{z} = \sum_{m=1}^{M_{\text{use}}} w_m z_m \Bigg/ \sum_{m=1}^{M_{\text{use}}} w_m$$

$$\sigma_z = \left[\sum_{m=1}^{M_{use}} w_m (z_m - \bar{z})^2 \Bigg/ \sum_{m=1}^{M_{use}} w_m \right]^{\frac{1}{2}}$$

For the calculation of the first numerical standard error τ_0, we assume no serial correlation in $(z_1, \ldots, z_M)$. This is appropriate for independence or importance sampling. Following Geweke (1989) [3], this leads to

$$\tau \approx \tau_0 \equiv \left[\sum_{m=1}^{M_{use}} w_m^2 (z_m - \bar{z})^2 \Bigg/ (\sum_{m=1}^{M_{use}} w_m)^2 \right]^{\frac{1}{2}}$$

For the calculation of τ_1 through τ_K, the remaining K estimates of the numerical standard error, the following method is used. First, `expect1` calculates group and sample means of the numerator quantity $w_m z_m$ and the denominator quantity w_m:

$$n(t) = \frac{1}{J} \sum_{m=(t-1)J+1}^{tJ} w_m z_m \qquad d(t) = \frac{1}{J} \sum_{m=(t-1)J+1}^{tJ} w_m \qquad t = 1, \ldots, T$$

$$\bar{n} = \frac{1}{M_{\text{use}}} \sum_{m=1}^{M_{\text{use}}} w_m z_m \qquad \bar{d} = \frac{1}{M_{\text{use}}} \sum_{m=1}^{M_{\text{use}}} w_m$$

Then it calculates the following sample autocorellation and autcovariance functions:

$$\gamma_{nn}(t) = \frac{1}{T} \sum_{s=t+1}^{T} (n(s) - \bar{n})(n(s-t) - \bar{n}) \qquad t = 0, \ldots, T-1$$

$$\gamma_{dd}(t) = \frac{1}{T} \sum_{s=t+1}^{T} (d(s) - \bar{d})(d(s-t) - \bar{d}) \qquad t = 0, \ldots, T-1$$

$$\gamma_{nd}(t) = \frac{1}{T} \sum_{s=t+1}^{T} (n(s) - \bar{n})(d(s-t) - \bar{d}) \qquad t = 0, \ldots, T-1$$

$$\gamma_{dn}(t) = \frac{1}{T} \sum_{s=t+1}^{T} (d(s) - \bar{d})(n(s-t) - \bar{n}) \qquad t = 0, \ldots, T-1$$

Then it calculates, for each $k \in \{1, \ldots, K\}$, estimates $\sigma^2_{n(k)}$, $\sigma^2_{d(k)}$, and $\sigma_{nd(k)}$ of $\sigma^2_n \equiv \mathrm{V}ar[n(t)]$, $\sigma^2_d \equiv \mathrm{V}ar[d(t)]$ and $\sigma_{nd} \equiv \mathrm{C}ov[n(t), d(t)]$, based on the taper half-width λ_k:

$$\sigma^2_{nn(k)} = \gamma_{nn}(0) + 2 \sum_{s=1}^{\lambda_k - 1} \frac{\lambda_k - s}{\lambda_k} \gamma_{nn}(s)$$

$$\sigma^2_{dd(k)} = \gamma_{dd}(0) + 2 \sum_{s=1}^{\lambda_k - 1} \frac{\lambda_k - s}{\lambda_k} \gamma_{dd}(s)$$

$$\sigma^2_{nd(k)} = \gamma_{nd}(0) + \sum_{s=1}^{\lambda_k - 1} \frac{\lambda_k - s}{\lambda_k} \left[\gamma_{nd}(s) + \gamma_{dn}(s)\right]$$

These calculations are based on conventional time series methods for a wide sense stationary process, described in Geweke (1992) [4].

By the conventional asymptotic expansion, the square of the numerical standard error is approximated by

$$\tau^2 = \mathrm{V}ar(\frac{n}{d}) \approx \left[\begin{array}{cc} \frac{1}{d} & -\frac{1}{d^2} \end{array} \right] \left[\begin{array}{cc} \sigma^2_n & \sigma_{nd} \\ \sigma_{nd} & \sigma^2_d \end{array} \right] \left[\begin{array}{c} \frac{1}{d} \\ -\frac{1}{d^2} \end{array} \right]$$

For each $k \in \{1, \ldots, K\}$ it calculates the approximation τ_k using $\sigma^2_{nn(k)}$, $\sigma^2_{dd(k)}$ and $\sigma^2_{nd(k)}$ defined above.

Relative numerical efficiencies $(\nu_0, \ldots, \nu_K)$ are calculated using

$$\nu_k \equiv \left(\frac{\tau_0}{\tau_k}\right)^2 \qquad k = 0, \ldots, K$$

3.3.3 The `expectN` Command

Description

Calculates combined sample means, with numerical standard errors, for a set of different weighted random samples, and tests for the equality of their individual population means.

Usage

```
[mean, nse, equal] = expectN(logWeight1, sample1, logWeight2, sample2);
```

```
[mean, nse, equal] = expectN(logWeight1, sample1, logWeight2, sample2,
taper);
```

Inputs

`logWeight1`	Vector of length M_1: log sample weights for first sample
`sample1`	Vector of length M_1: first sample of scalar draws
`logWeight2`	Vector of length M_2: log sample weights for second sample
`sample2`	Vector of length M_2: second sample of scalar draws
`taper`	Vector of length K: taper half-widths (optional)

Outputs

`mean`	Vector of length $K + 1$: estimated combined weighted sample means
`nse`	Vector of length $K + 1$: estimated numerical standard errors
`equal`	Vector of length $K + 1$: marginal significance levels for a chi-squared test of the equality of the population means

See Also

`expect1`.

Example

```
% Use default taper values
[mean, nse, p] = expectN(lw1, z1, lw2, z2);
% Use alternate taper values
taper = [4.0,8.0];
[mean, nse, p] = expectN(lw1, z1, lw2, z2, taper);
```

Details

In general, there are N pairs of weighted samples, not just two. For each sample $z^{(i)}$, `expectN` calculates individual sample moments $\bar{z}^{(i)}$ and estimates of numerical standard errors $(\tau_0^{(i)}, \ldots, \tau_K^{(i)})$ from the samples $(z_1^{(i)}, \ldots, z_{M_i}^{(i)})$, the log weights $(\log z_1^{(i)}, \ldots, \log z_{M_i}^{(i)})$, and the half-taper values $(\lambda_1, \ldots, \lambda_K)$, in the same way that `expect1` calculates $\bar{z}$ and $(\tau_0, \ldots, \tau_K)$ from $(z_1, \ldots, z_M)$, $(\log z_1, \ldots, \log z_M)$, and $(\lambda_1, \ldots, \lambda_K)$.

The estimated sample means $\bar{z}_k$ are given by

$$\bar{z}_k = \sum_{i=1}^{N} \frac{\bar{z}^{(i)}}{\tau_k^{2(i)}} \Bigg/ \sum_{i=1}^{N} \frac{1}{\tau_k^{2(i)}} \qquad k = 1, \ldots, K$$

The estimated numerical standard errors τ_k are given by

$$\frac{1}{\tau_k^2} = \sum_{i=1}^{N} \frac{1}{\tau_k^{2(i)}}$$

For each k, the marginal significance level is the value of p_k such that

$$\left[\begin{array}{ccc} \bar{z}_k^{(2)} - \bar{z}_k^{(1)} & \cdots & \bar{z}_k^{(N)} - \bar{z}_k^{(N-1)} \end{array} \right] \cdot \Sigma^{-1} \cdot \left[\begin{array}{c} \bar{z}_k^{(2)} - \bar{z}_k^{(1)} \\ \bar{z}_k^{(3)} - \bar{z}_k^{(2)} \\ \vdots \\ \bar{z}_k^{(N)} - \bar{z}_k^{(N-1)} \end{array} \right] = \chi^2_{1-p_k}(N-1)$$

where Σ is the following matrix

$$\left[\begin{array}{cccccc} \tau_k^{2(1)} + \tau_k^{2(2)} & -\tau_k^{2(2)} & 0 & \cdots & 0 & 0 \\ -\tau_k^{2(2)} & \tau_k^{2(2)} + \tau_k^{2(3)} & -\tau_k^{2(3)} & \cdots & 0 & 0 \\ 0 & -\tau_k^{2(3)} & \tau_k^{2(3)} + \tau_k^{2(4)} & \cdots & 0 & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & 0 & \ldots & \tau_k^{2(N-2)} + \tau_k^{2(N-1)} & -\tau_k^{2(N-1)} \\ 0 & 0 & 0 & \ldots & -\tau_k^{2(N-1)} & \tau_k^{2(N-2)} + \tau_k^{2(N-1)} \end{array} \right]$$

3.3.4 The `extract` Command

Description

Returns simulation matrices for a model instance.

Usage

```
sim = extract(modelInst);
```

Inputs

`modelInst`	Integer: model instance identifier.

Outputs

`sim`	Structure: simulation matrices

Example

```
sim = extract(mi);
```

Details

The return value is a structure (named list in S-PLUS and R) with the following fields (components in S-PLUS and R):

id Model instance identifier.

logWeightPost Log weights for posterior draws.

logPrior Value of log prior for posterior draws.

logXPrior Value of transformed log prior values for posterior draws

logLike Value of log likelihood for posterior draws.

logPriorHM Value of log prior for posterior HM draws.

logLikeHM Value of log likelihood for posterior HM draws.

logWeightPrior Log weights for prior draws.

logPriorPrior Value of log prior for prior draws.

logLikePrior Value of log likelihood for prior draws.

***** Posterior simulation matrix of unknown quantity named *.

***Prior** Prior simulation matrix of unknown quantity named *.

***HM** Posterior HM simulation matrix of unknown quantity named *.

All simulation matrices have three dimensions. The first two dimensions give the row and column of the unknown quantity. The third dimension is the simulation dimension. Each value of the third index gives a different draw of the unknown quantity.

3.3.5 The `gammaSim` Command

Description

Generates a sample from a gamma distribution.

Usage

```
sample = gammaSim(alpha, beta, n);
```

Inputs

`alpha`	Real scalar: shape parameter of gamma distribution
`beta`	Real scalar: inverse scale parameter of gamma distribution
`n`	Integer: number of draws to generate

Outputs

`sample`	n by 1 matrix: sample generated from gamma distribution

See Also

`paretoSim`, `gaussianSim`, `dirichletSim`, `wishartSim`.

Example

```
alpha = 3.0;

beta = 5.0;

sample = gammaSim(alpha, beta, 1000);
```

Details

Each of the n draws of the sample is a scalar with a gamma distribution. The result is given as a n by 1 matrix.

See Appendix A for the parametrization of the gamma distribution.

3.3.6 The `gaussianSim` Command

Description

Generates a sample from a Gaussian distribution.

Usage

```
sample = gaussianSim(mean, precision, n);
```

Inputs

`mean`	Vector of length K: mean of Gaussian distribution
`precision`	K by K matrix: precision of Gaussian distribution
`n`	Integer: number of draws to generate

Outputs

`sample`	n by K matrix: sample generated from Gaussian distribution

See Also

`paretoSim`, `dirichletSim`, `gammaSim`, `wishartSim`.

Example

```
mean = [1.0;2.0];

precision = [1.0,0.0;0.0,1.0];

sample = gaussianSim(mean, precision, 1000);
```

Details

Each of the n draws of the sample is a vector of length K with a Gaussian distribution. The result is given as a n by K matrix.

See Appendix A for the parametrization of the Gaussian distribution.

3.3.7 The `listModelSpecs` Command

Description

Lists all available model specifications (e.g. nlm, poisson).

Usage

```
listModelSpecs;
```

Inputs

None.

Outputs

None.

See Also

`minst`, `listModels`.

Example

```
listModelSpecs;
```

Details

A printed message gives a list of model specifications.

3.3.8 The `listModels` Command

Description

Lists all open model instances.

Usage

```
listModels;
```

Inputs

None.

Outputs

None.

See Also

`minst`, `miDelete`, `listModelSpecs`.

Example

```
listModels;
```

Details

A printed message gives the model instance identification number, the name of the model specification (e.g. nlm, poisson), and the number of prior, posterior, and posterior HM draws.

3.3.9 The `miDelete` Command

Description

Closes without saving a (or all) model instances.

Usage

```
miDelete(modelInst);
```

Inputs

`modelInst` Integer: model instance identifier.

Outputs

None.

See Also

`minst`, `listModels`.

Example

```
miDelete(mi);
```

3.3.10 The `miLoad` Command

Description

Loads a model instance stored in a binary file.

Usage

```
modelInst = miLoad(filename);
```

Inputs

`filename`	String: name of binary file storing the model instance

Outputs

`modelInst`	Integer: model instance identifier.

See Also

`miSave`, `minst`, `miLoadAscii`.

Example

```
mi = miLoad("miFile");
```

3.3.11 The `miLoadAscii` Command

Description

Loads a model instance stored in a text file.

Usage

```
modelInst = miLoadAscii(filename);
```

Inputs

`filename`	String: name of text file storing the model instance

Outputs

`modelInst`	Integer: model instance identifier.

See Also

`miSaveAscii`, `minst`, `miLoad`.

Example

```
mi = miLoadAscii("miFile.txt");
```

3.3.12 The `miSave` Command

Description

Saves a model instance in a binary file.

Usage

```
miSave(modelInst, filename);
```

Inputs

`modelInst`	Integer: model instance identifier.
`filename`	String: name of binary file in which to store the model instance

Outputs

None.

See Also

`miLoad`, `minst`, `miSaveAscii`.

Example

```
miSave(mi, "miFile");
```

Details

If the file already exists, it is written over.

3.3.13 The `miSaveAscii` Command

Description

Saves a model instance in a text file.

Usage

```
miSaveAscii(modelInst, filename);
```

Inputs

`modelInst`	Integer: model instance identifier.
`filename`	String: name of text file in which to store the model instance

Outputs

None.

See Also

`miLoadAscii`, `minst`, `miSave`.

Example

```
miSaveAscii(mi, "miFile.txt");
```

Details

If the file already exists, it is written over. The ascii version of a model instance is platform independent and human readable, but long and inefficient.

3.3.14 The `minst` Command

Description

Creates an instance of a particular model specification.

Usage

```
modelInst = minst(modelSpecName, unknownNames, knowns);
```

Inputs

`modelSpecName`	String: name of model specification
`unknownNames`	List of strings separated by commas: user provided names for unknown quantities
`knowns`	List of matrices separated by commas: user provided matrices of known quantities

Outputs

`modelInst`	Integer: model instance identifier.

Example

```
a = [1,1,2];

s = [1,1;2,3;3,3];

myMI = minst('iidfs', 'pi', a, s);
```

Details

The available model specifications are described in Chapter 2. For each model specification, there is a section "Creating a Model Instance" with the relevant information, namely

- The name of the model specification.
- The order in which the user specifies the names for the unknown quantities of the model.
- The order in which the user provides the matrices giving the values of the known quantities of the model.

3.3.15 The `mlike` Command

Description

Computes various estimates of the marginal likelihood for a model instance, with numerical standard errors.

Usage

```
[ml, mlnse] = mlike(modelInst);
[ml, mlnse] = mlike(modelInst, p, taper);
```

Inputs

`modelInst`	Integer: model instance identifier.
`p`	Vector of length L: truncation parameters (optional)
`taper`	Vector of length K: taper half-widths (optional)

Outputs

`ml`	Vector of length L: marginal likelihood estimates
`mlnse`	L by $K+1$ matrix: numerical standared error estimates

See Also

`postsim`, `postsimHM`.

Example

```
% Use default truncation and taper values
[ml, mlNSE, mlHM, mlNSEHM] = mlike(mi);
% Use alternate truncation values
p = [0.1,0.3,0.5,0.7,0.9];
[ml, mlNSE, mlHM, mlNSEHM] = mlike(mi, p);
% Use alternate truncation and taper values
taper = [4.0,8.0];
[ml, mlNSE, mlHM, mlNSEHM] = mlike(mi, p, taper);
```

Details

The method used is a modification described in Geweke [5] of the method proposed in Gelfand and Dey [2].

The truncation parameters $p_l \in [0,1]$ index the truncated multivariate normal distribution $f(\cdot)$ discussed in Geweke [5]. For each p_l, `mlike` generates (internally) an unweighted vector $(z_1^l, \ldots, z_M^l)$, where M is the number of posterior samples in the given model instance.

For each l, `mlike` calculates the sample mean $\bar{z}_l$ and numerical standard errors $(\tilde{\tau}_1^l, \ldots, \tilde{\tau}_M^l)$ from $(z_1^l, \ldots, z_M^l)$ and $(\lambda_1, \ldots, \lambda_K)$ in the same way that `expect1` calculates $(\tau_0, \ldots, \tau_K)$ from $(z_1, \ldots, z_M)$, a vector of equal log weights, and $(\lambda_1, \ldots, \lambda_K)$. Then for all l, the estimate of the log marginal likelihood is given by

$$\mu_l = -\log \bar{z}_l$$

and for all l and k, the estimate of the numerical standard error for the log marginal likelihood is given by

$$\tau_{kl} = \frac{\tilde{\tau}_{kl}}{\bar{z}_l}$$

When numerical standard error is small, results are not sensitive to the choice of p. In these cases $L = 1$ and $p_1 = 0.5$ will suffice. However the additional computational burden of increasing L is negligible. If you are concerned about standard errors, it is best to use several values of p_l, for example, $p = (0.1, 0.2, \ldots, 0.9)$.

3.3.16 The `paretoSim` Command

Description

Generates a sample from a Pareto distribution.

Usage

```
sample = paretoSim(alpha, beta, n);
```

Inputs

`alpha`	Real scalar: tail parameter of Pareto distribution
`beta`	Real scalar: location parameter of Pareto distribution
`n`	Integer: number of draws to generate

Outputs

`sample`	n by 1 matrix: sample generated from Pareto distribution

See Also

`dirichletSim`, `gaussianSim`, `gammaSim`, `wishartSim`.

Example

```
alpha = 1.0;

beta = 4.0;

sample = gammaSim(alpha, beta, 1000);
```

Details

Each of the n draws of the sample is a scalar with a pareto distribution. The result is given as a n by 1 matrix.

See Appendix A for the parametrization of the Pareto distribution.

3.3.17 The `postfilter` Command

Description

Filters out previously generated draws from the posterior simulation matrix of a given model instance.

Usage

```
postfilter(modelInst, filter);
```

Inputs

`modelInst`	Integer: model instance identifier.
`filter`	Vector of integers of length n: indices of existing draws to keep

Outputs

None.

Example

```
filter = [101:1000];

postfilter(mi, filter);
```

Details

The ith draw of the posterior simulation matrix is kept if and only if $i = f_j$ for some j from 1 to n.

3.3.18 The `postsim` Command

Description

Generates or appends to the posterior simulation matrix of a given model instance.

Usage

```
postsim(modelInst, m, n);
```

Inputs

`modelInst`	Integer: model instance identifier.
`m`	Integer: number of posterior draws to record
`n`	Integer: number of posterior draws to generate for each one recorded

Outputs

None.

See Also

`minst`, `postfilter`, `mlike`, `priorsim`, `postsimHM`, `extract`.

Example

```
postsim(mi, 1000, 1);
```

Details

Generates draws of unknown quantities from their posterior distribution. Generates mn new posterior draws, and appends every nth draw to the posterior simulation matrix. If there are any draws from a previous invocation of `postsim`, the first new draw comes from the transition kernel of the Markov chain used for posterior simulation. Otherwise, it comes from the initial distribution of the Markov chain.

Use the `extract` command to obtain the posterior draws.

3.3.19 The `postsimHM` Command

Description

Generates or appends to the posterior HM simulation matrix of a given model instance.

Usage

```
postsimHM(modelInst, m, n, scalePrecision);
```

Inputs

`modelInst`	Integer: model instance identifier.
`m`	Integer: number of posterior draws to record
`n`	Integer: number of posterior draws to generate for each one recorded
`scalePrecision`	Real scalar: factor used to rescale the precision matrix of the random walk innovation

Outputs

None.

See Also

`minst`, `mlike`, `postsim`, `extract`.

Example

```
postsimHM(mi, 1000, 1, 10.0);
```

Details

Generates draws of unknown quantities from their posterior distribution using a Gaussian random walk Metropolis chain with proposal covariance proportional to the sample covariance of draws from the posterior simulation matrix. Generates mn new posterior draws, and appends every nth draw to the posterior simulation matrix. If there are any draws from a previous invocation of `postsimHM`, the first new draw comes from the transition kernel of the Markov chain used for posterior simulation. Otherwise, it comes from the initial distribution of the Markov chain.

Use the `extract` command to obtain the posterior HM draws.

3.3.20 The `priorRobust` Command

Description

Calculates upper and lower bounds on the mean of a posterior function of interest, as the prior distribution is varied from its original specification.

Usage

```
[mean, std, U, L, Ut, Lt] = priorRobust(logWeight, sample, factors);
```

Inputs

`logWeight`	Vector of length m: log weights
`sample`	Vector of length m: posterior sample of some scalar function of interest
`factors`	Vector of length n: bound factors for robustness analysis

Outputs

`mean`	Real scalar: posterior sample mean for original prior specification
`std`	Real scalar: posterior sample standard deviation for original prior specification
`U`	Vector of length n: exact upper bounds
`L`	Vector of length n: exact lower bounds
`Ut`	Vector of length n: asymptotic upper bounds
`Lt`	Vector of length n: asymptotic lower bounds

Example

```
K = [5.0,10.0,20.0];
[mean, std, U, L, Ut, Lt] = priorRobust(lw, beta, K);
```

Details

For each bound factor, calculates exact lower and upper bounds and asymptotic lower and upper bounds for the posterior mean. For each bound parameter k_i, `priorRobust` calculates exact lower and upper bounds L_i and U_i for the posterior mean of the function of interest g, for the following set of prior density kernels.

$$\left\{ p^*(\cdot) : \frac{1}{k_i} p(\theta) \leq p^*(\theta) \leq k_i p(\theta) \quad \forall \theta \in \Theta \right\}$$

where $p(\cdot)$ is the actual prior density. It uses the algorithm described in Geweke and Petrella [6]. Also for each k_i, `priorRobust` calculates asymptotically valid

lower and upper bounds $\tilde{L}_i$ and $\tilde{U}_i$, using the results of DeRobertis and Hartigan [1].

3.3.21 The `priorfilter` Command

Description

Filters out previously generated draws from the prior simulation matrix of a given model instance.

Usage

```
priorfilter(modelInst, filter);
```

Inputs

`modelInst`	Integer: model instance identifier.
`filter`	Vector of integers of length n: indices of existing draws to keep

Outputs

None.

Example

```
filter = [101:1000];

priorfilter(mi, filter);
```

Details

The ith draw of the prior simulation matrix is kept if and only if $i = f_j$ for some j from 1 to n.

3.3.22 The `priorsim` Command

Description

Generates or appends to the prior simulation matrix of a given model instance.

Usage

```
priorsim(modelInst, m, n);
```

Inputs

`modelInst`	Integer: model instance identifier.
`m`	Integer: number of prior draws to record
`n`	Integer: number of prior draws to generate for each one recorded

Outputs

None.

See Also

`minst`, `priorfilter`, `postsim`, `extract`.

Example

```
priorsim(mi, 1000, 1);
```

Details

Generates draws of unknown quantities from their prior distribution. Generates mn new prior draws, and appends every nth draw to the prior simulation matrix. If there are any draws from a previous invocation of `priorsim`, the first new draw comes from the transition kernel of the Markov chain used for prior simulation. Otherwise, it comes from the initial distribution of the Markov chain.

Use the `extract` command to obtain the prior draws.

3.3.23 The `setseedconstant` Command

Description

Sets the seeds of the random number generators to a constant value.

Usage

```
setseedconstant;
```

Inputs

None.

Outputs

None.

See Also

`setseedtime`.

Example

```
setseedconstant;
```

Details

This is useful for ensuring that repeated invocations of a command generating random values lead to the same results.

3.3.24 The `setseedtime` Command

Description

Sets the seeds of the random number generators to the number of seconds since the beginning of 1970.

Usage

```
setseedtime;
```

Inputs

None.

Outputs

None.

See Also

`setseedconstant`.

Example

```
setseedtime;
```

Details

This is useful for ensuring that repeated invocations of a command generating random values lead to different results.

3.3.25 The `weightedSmooth` Command

Description

Estimates a univariate density function for a weighted random sample, using a kernel smoothing algorithm adapted to weighted samples.

Usage

```
[x, y] = weightedSmooth(logWeight, sample);
```

Inputs

`logWeight`	Vector of length M: log weights
`sample`	Vector of length M: a posterior sample of some function of interest
`ktype`	String: kernel type (optional)
`krange`	String: kernel range type (optional)
`wwf`	Real scalar: window width fraction (optional)
`nplot`	Integer: number of ordered pairs to generate (optional)
`range_a1`	Real scalar: left bound range parameter (optional)
`range_a2`	Real scalar: right bound range parameter (optional)

Outputs

`x`	Vector of length N: ordinate values
`y`	Vector of length N: abscissa values

Example

```
[x, y] = weightedSmooth(lw, z);
nplot = 2000;
ktype = triangular;
[x, y] = weightedSmooth(lw, z);
```

Details

The estimated density at a point z is

$$f(z) = \frac{\sum_{m=1}^{M} w_m K\left(\frac{z-z_m}{h}\right)}{h \sum_{m=1}^{M} w_m}$$

The functional form of the kernel function K depends on the value of `ktype` according to Table 3.1.

Table 3.1: Values of `Ktype`

`Ktype`	K
`uniform`	$K(t) = \frac{1}{2}\chi_{(-1,1)}(t)$
`triangle`	$K(t) = (1 - \|t\|)\chi_{(-1,1)}(t)$
`biweight`	$K(t) = \frac{15}{16}(1 - z^2)^2\chi_{(-1,1)}(t)$

For any set S, the function $\chi_S(\cdot)$ is a set membership indicator function.

The value h is given by

$$h = \lambda(q_{\frac{3}{4}} - q_{\frac{1}{4}})$$

where q_α denotes the α'th sample quantile of z.

The `weightedSmooth` command generates N ordered pairs (x_i, y_i). The values x_i are evenly spaced between x_{min} and x_{max}, determined by `Krange` according to Table 3.2. The values y_i satisfy $y_i = f(x_i)$.

Table 3.2: Values of `Krange`

`Krange`	x_{min}	x_{max}
`quantile`	q_{a_1}	q_{a_2}
`absolute`	a_1	a_2

For most plotting routines, N should be in the range of 200 to 400. The choice of λ depends on how smooth the resulting plot is desired to be. As with all kernel smoothing methods, some experimentation will probably be necessary. The greater the number of simulations available, the smaller λ can be and still retain visual smoothness. It is generally easier to use the `Krange=quantile` option and specify a_1 in the range .001 to .01 and a_2 in the range .99 to .999; this will include the important part of the estimated density while not wasting space on the plot for points where the density is small.

3.3.26 The `wishartSim` Command

Description

Generates a sample from a Wishart distribution.

Usage

```
sample = wishartSim(A, nu, n);
```

Inputs

`A`	m by m matrix: inverse scale parameter of Wishart distribution
`nu`	Real scalar: degrees of freedom parameter of Wishart distribution
`n`	Integer: number of draws to generate

Outputs

`sample`	n by m^2 matrix: sample generated from Wishart distribution

See Also

`paretoSim`, `gaussianSim`, `gammaSim`, `dirichletSim`.

Example

```
A = [1.0,0.0;0.0,1.0];
nu = 100;
sample = wishartSim(A, nu, 1000);
```

Details

Each of the n draws of the sample is an m by m matrix with a Wishart distribution. The result is given as a m by m^2 matrix.

See Appendix A for the parametrization of the Wishart distribution.

Chapter 4

A BACC Tutorial

In order to answer commonly asked questions, this chapter contains a step-by-step tutorial with explanations of what each step is doing and what each term means.

4.1 Loading Known Quantities

To begin using BACC you first need to specify the known quantities. These quantities can be entered directly into Matlab, loaded from a Matlab file, or loaded from a text file. The next lines load the prior matrices and data matrices for the normal linear model. The files for this example are in the `test` directory contained in the BACC zipfile for Matlab users.

```
load betahd.mtx;
load Hhd.mtx;
load nuhd.mtx;
load shd.mtx;
load Xhd.mtx;
load yhd.mtx;
```

4.2 Creating a Model Instance

Next, a "model instance" is created. A model instance can be though of as a box containing everying that BACC needs to know about a particular model. This includes the type of the model, i.e., normal linear model, a list of names for the knowns, and the names of the known quantities. The next line creates an instance of the normal linear model, assigns a model identification number to it, and places this number in `mi`. References to `mi` now refer to this model instance.

```
mi=minst('nlm','beta','h',betahd,Hhd,nuhd,shd,Xhd,yhd);
```

To see a list of available model specifications (e.g., `nlm`) that can be used with `minst` type

```
listModelSpecs
```

4.3 Simulating the Model

Simulation can be performed from the prior or the posterior of a model using the commands `priorsim` and `postsim`. After posterior simulation has been done, it is also possible to simulate the posterior using a random walk Hastings-Metropolis algorithm. This is done using the command `postsimhm`. If you wish to compare output across runs, it is a good idea to set the seeds of the random number generators. This is done using the `setSeedConstant` command. Strictly speaking, this sets the seeds of another routine that then sets the seeds of the random number generators.

First, generate 5000 samples from the prior.

```
setSeedConstant;
priorsim(mi,5000,1);
```

Now generate 1000 samples from the posterior.

```
setSeedConstant;
postsim(mi,1000,1);
```

To filter out posterior samples you can specify the numbers of the samples you wish to keep. For example, to drop the first 100 samples you would create a vector containing the numbers 101 to 1000.

```
filt=[101:1000];
```

Then pass this list of samples you wish to keep to the `postfilter` routine.

```
postfilter(mi,filt);
```

Now you are left with 900 posterior samples. Add another 4100 samples.

```
setSeedConstant;
postsim(mi,4100,1);
```

We can also simulate 10000 samples from the posterior using a random walk Hastings-Metropolis algorithm.

```
setSeedConstant;
postsimhm(mi,10000,1,1);
```

4.4 Extracting and Deleting Model Instances

To manipulate a model instance in the Matlab workspace, it needs to be extracted from memory. Type

```
mysim=extract(mi)
```

to extract the model instance associated with the identification number `mi` to a Matlab structure called `mysim`. Alternatively, this can be done automatically following simulation by typing

```
mysim=postsim(mi,4100,1);
```

in place of

```
postsim(mi,4100,1);
```

Similarly with `priorsim` and `postsimhm`.

To see a list of your current model instances (in memory, not necessarily in the Matlab workspace) type

```
listModels
```

To delete the model instance associated with `mi` from memory, type

```
deleteModel(mi)
```

To delete *all* model instances in memory, type

```
deleteModel
```

4.5 Computing Marginal likelihoods

The `mlike` command can report several log marginal likelihood values for a model instance. First, if the model has an analytic marginal likelihood, then the exact marginal likehood is printed. Then, for either a default truncation parameter or for a user specified vector of truncation paramters, `mlike` computes an estimate of the log marginal likelihood for each truncation parameter. Finally, if there are `postsimhm` samples as well, then `mlike` also reports estimates of the log marginal likelihood based on those samples. To use the default truncation parameter and compute the log marginal likelihoods type

```
mlike(mi);
```

or to specify a vector of truncation parameters type (for example)

```
p=[0.1 0.2 0.3 0.4 0.5 0.6 0.7 0.8 0.9];
mlike(mi,p);
```

4.6 Computing Moments of Functions of Interest

To compute the posterior mean of the first element of `beta` type

```
expect1(mysim.logweight,mysim.beta(1,1,:));
```

To specify taper parameters rather than the use the default values, type

```
taper=[4.0 8.0];
expect1(mysim.logweight,mysim.beta(1,1,:),taper);
```

More generally, `expect1` can compute the mean and standard deviation of any function of interest. The first argument is a vector of log weight values and the second argument is a vector of function of interest values. In some cases it may be sufficient to simply change the weight vector. For example, to find the mean of `h` conditional on `h` being greater than 0.8 the log weight vector could be defined as:

```
lw = log( mysim.h > .8 );
```

If `h` is greater than 0.8, then `lw` equals `log(1) = 0` for that sample. If `h` is less than or equal to 0.8, then `lw` equals `log(0)=-Infinity` for that sample. Hence, the mean of the prior samples of `h` conditional on `h` being greater than 0.8 is found by

```
expect1(lw,mysim.h);
```

In other cases the second argument might be changed. For example,

```
foi = 1./mysim.h;
expect1(mysim.logweight,foi);
```

To compute combined sample means and test for the equality of the individual population means use the `expectN` command. For example, to compare the posterior and prior means for the first element of the `beta` vector, type

```
expectN(mysim.logweight,mysim.beta(1,1,:),mysim.logweight,mysim.betaPrior(1,1,:),taper);
```

4.7 Sensitivity to the Prior

To calculate upper and lower bounds on the mean of a posterior function of interest, as the prior distribution is varied from its original specification type

```
priorRobust(mysim.logweight,mysim.beta(1,1,:));
```

To specify values of the `bounds` parameter other than the default values,

```
bounds=[2 10];
priorRobust(mysim.logweight,mysim.beta(1,1,:),bounds);
```

4.8 Kernel Smoothing of Simulations

To estimate a univariate density function for a weighted random sample, using kernel smoothing, type

```
[x,y]=weightedSmooth(mysim.logweight,mysim.beta(1,1,:));
```

Then plot the kernel using the Matlab plot command,

```
plot(x,y);
```

The `weightedSmooth` command has various settings that can be changed. For example,

to generate more samples and use a different kernel you might type

```
nplot=2000;
ktype='triangular';
[x,y]=weightedSmooth(mysim.logweight,mysim.beta(1,1,:));
```

4.9 Plotting

The user can use all of the standard Matlab plotting tools. Before plotting unknown quantities, however, they must be converted from 3-dimensional arrays to 2-dimensional arrays. The first two dimensions for an unknown quantity refer to the actual parameter dimensions. The third dimension is the sample index. Conversions can be done using the Matlab command `squeeze`. The command removes any singleton dimensions. To convert the first `beta` parameter into a two dimensional array type

```
beta1=squeeze(mysim.beta(1,1,:));
```

To convert all of the `beta`'s into an $M \times k$ array, where M indexes the number of samples and k is the number of elements of `beta`, type

```
beta=squeeze(mysim.beta(:,1,:))';
```

With an understanding of `squeeze` it is now possible to discuss various plotting tools.

For example, to get a histogram of the posterior simulation of the unknown parameter `h`, type

```
hist(mysim.h);
```

or

```
hist(mysim.h,40);
```

where the second command increases the number of bins used in the histogram.

To plot a "time-series" of the loglikelihood values, type

```
plot(mysim.loglike);
```

To plot a scatterplot of the third and fourth elements of `beta`, type

```
scatter(beta(:,3),beta(:,4));
```

where `beta` refers to the `squeeze`'d matrix from above.

4.10 Saving and Loading Model Instances

4.10.1 To/From Text Files

To save a model instance called `mi` to an ascii file called `mi.txt` use the command

```
miSaveAscii(mi,'mi.txt');
```

To load a model instance stored in an ascii file called `mi.txt` and to give it the name `mi`, use the command

```
mi=miLoadAscii('mi.txt');
```

4.10.2 To/From Binary Files

To save a model instance called `mi` to a binary file called `mi.bin` use the command

```
miSave(mi,'mi.bin');
```

To load a model instance stored in a binary file called `mi.bin` and to give it the name `mi`, use the command

```
mi=miLoad('mi.bin');
```

4.11 Drawing from Various Distributions

The following code demonstrates how to draw from various distributions. See Appendix A for the parameterization of these distributions.

4.11.1 Dirichlet

```
a=[1 2 3; 4 5 6];
setSeedConstant;
sample=dirichletSim(a,1000);
disp('dirichlet mean');
disp(sum(sample)/1000);
```

4.11.2 Gamma

```
setSeedConstant;
sample=gammaSim(3,5,1000);
disp('gamma mean');
disp(sum(sample)/1000);
```

4.11.3 Gaussian (Multivariate Normal)

```
mean=[1;2];
precision = [1 0; 0 1];
setSeedConstant;
sample=gaussianSim(mean,precision,1000);
disp('gaussian mean');
disp(sum(sample)/1000);
```

4.11.4 Pareto

```
setSeedConstant;
sample=paretoSim(3,5,1000);
disp('pareto mean');
disp(sum(sample)/1000);
```

4.11.5 Wishart (Multivariate Chi-squared)

```
scale=precision;
setSeedConstant;
sample=wishartSim(scale,10,1000);
disp('wishart mean');
disp(sum(sample)/1000);
```

Appendix A

Distributions

This appendix gives the density and mass functions for the distributions used in this document.

A.1 The Dirichlet Distribution

A random vector π of length n has the Dirichlet distribution with parameter vector $\alpha \in \Re_+^n$, denoted $\pi \sim \mathrm{Di}(\alpha)$, if its probability density function is

$$p(\pi|\alpha) = \begin{cases} m^{-1/2} \frac{\Gamma(\sum_{i=1}^m \alpha_i)}{\prod_{i=1}^m \Gamma(\alpha_i)} \prod_{i=1}^m \pi_i^{\alpha_i - 1} & \pi \in \Delta_n \equiv \{p \in \Re_+^n : \sum_{i=1}^n p_i = 1\} \\ 0 & \text{otherwise} \end{cases}$$

The mean and variance are given by

$$\mathrm{E}[\pi_i|\alpha] = \frac{\alpha_i}{\sum_{j=1}^n \alpha_j}$$

$$\mathrm{V}ar[\pi_i|\alpha] = \frac{\mathrm{E}[\pi_i|\alpha](1 - \mathrm{E}[\pi_i|\alpha])}{1 + \sum_{j=1}^n \alpha_j}$$

$$\mathrm{C}ov[\pi_i, \pi_j|\alpha] = \frac{-\mathrm{E}[\pi_i|\alpha]\mathrm{E}[\pi_j|\alpha]}{1 + \sum_{k=1}^n \alpha_k}$$

A.2 The Gamma Distribution

A random scalar λ has the Gamma distribution with shape parameter $\alpha > 0$ and scale parameter $\beta > 0$, denoted $\lambda \sim \mathrm{Ga}(\underline{\alpha}, \underline{\beta})$, if its probability density function is

$$p(\lambda|\alpha, \beta) = \begin{cases} \frac{\beta^\alpha}{\Gamma(\alpha)} \lambda^{\alpha - 1} e^{-\beta\lambda} & \lambda > 0 \\ 0 & \text{otherwise} \end{cases}$$

The mean and variance are given by

$$\mathrm{E}[\lambda|\alpha,\beta] = \frac{\alpha}{\beta}$$
$$\mathrm{V}ar[\lambda|\alpha,\beta] = \frac{\alpha}{\beta^2}$$

A.3 The Normal Distribution

A random vector x has the Normal Distribution with mean parameter vector $\mu \in \Re^k$ and positive definite $k \times k$ variance parameter matrix Σ, denoted $x \sim \mathrm{N}(\mu, \Sigma)$, if its probability density function is

$$p(x|\mu,\Sigma) = |\Sigma|^{-\frac{1}{2}}(2\pi)^{-\frac{k}{2}} \exp\left[-\frac{1}{2}(x-\mu)'\Sigma^{-1}(x-\mu)\right] \qquad \forall x \in \Re^n$$

The mean and variane are given by

$$\mathrm{E}[x|\mu,\Sigma] = \mu$$
$$\mathrm{V}ar[x|\mu,\Sigma] = \Sigma$$

A.4 The Pareto Distribution

A random scalar x has the Pareto Distribution with parameters $\alpha > 0$ and $\beta \geq 0$, denoted $\theta \sim \mathrm{Pa}(\alpha, \beta)$, if its probability density function is

$$p(\theta|\alpha,\beta) = \begin{cases} \alpha\beta^\alpha\theta^{-(\alpha+1)} & \theta \geq \beta \\ 0 & \text{otherwise} \end{cases}$$

The mean and variance are given by

$$\mathrm{E}[\theta|\alpha,\beta] = \frac{\alpha\beta}{\alpha-1}$$
$$\mathrm{V}ar[\theta|\alpha,\beta] = \frac{\alpha\beta^2}{(\alpha-1)^2(\alpha-2)}$$

A.5 The Poisson Distribution

A discrete random variable x has the Poisson distribution with mean parameter $\lambda > 0$, denoted $x \sim \mathrm{Po}(\lambda)$, if its probability mass function is

$$p(x) = \begin{cases} e^{-\lambda}\frac{\lambda^x}{x!} & x \in \{0, 1, \ldots\} \\ 0 & \text{otherwise} \end{cases}$$

The mean and variance are given by

$$\mathrm{E}[x|\lambda] = \lambda$$
$$\mathrm{V}ar[x|\lambda] = \lambda$$

A.6 The Wishart Distribution

An $m \times m$ random matrix H has the Wishart distribution with positive definite $m \times m$ scale parameter matrix A and degrees of freedom parameter $\nu > m$, denoted $H \sim \mathrm{W}i(A, \nu)$, if its probability density function is

$$p(H|A,\nu) = \begin{cases} \frac{\pi^{-m(m-1)/4}|A|^{-\nu/2}}{2^{m\nu/2}\prod_{i=1}^{m}\Gamma(\frac{\nu-i+1}{2})} \cdot |H|^{(\nu-m-1)/2}\exp\left[-\frac{1}{2}\mathrm{t}r(A^{-1}H)\right] & H \text{ p.d.} \\ 0 & \text{otherwise} \end{cases}$$

The mean, and mean of the matrix H^{-1} are given by

$$\mathrm{E}[H|A,\nu] = \nu A$$
$$\mathrm{E}[H^{-1}|A,\nu] = \frac{1}{\nu-m-1}A^{-1}$$

Bibliography

[1] DeRobertis, L., and J. A. Hartigan, 1981, "Bayesian Inference Using Intervals of Measures," *The Annals of Statistics* 9: 235-244.

[2] Gelfand, A.E. and D.K. Dey, 1994, "Bayesian Model Choice: Asymptotics and Exact Calculations," *Journal of the Royal Statistical Society Series B* **56**: 501-514.

[3] Geweke, J., 1989, "Bayesian Inference in Econometric Models Using Monte Carlo Integration" *Econometrica,* 57, 1317-1340.

[4] Geweke, J., 1992, "Evaluating the Accuracy of Sampling-Based Approaches to the Calculation of Posterior Moments," in J.O. Berger, J.M. Bernardo, A.P. Dawid, and A.F.M. Smith (eds.), *Proceedings of the Fourth Valencia International Meeting on Bayesian Statistics,* 169-194. Oxford: Oxford University Press.

[5] Geweke, J. 1999, "Simulation-Based Bayesian Inference for Economic Time Series," in R.S. Mariano, T. Schuermann and M. Weeks (eds.), *Simulation-Based Inference in Econometrics: Methods and Applications.* Cambridge: Cambridge University Press, forthcoming.

[6] Geweke, J. and L. Petrella, 1999, "Prior Density Ratio Class Robustness in Econometrics," Journal of Business and Economic Statistics, forthcoming.

[7] Raftery, A.E., 1995, "Hypothesis testing and model selection via posterior simulation," University of Washington working paper.